Toolkit IV

TURTLEGRAPHICS

USERS MANUAL

KYAN SOFTWARE INC.
SAN FRANCISCO, CALIFORNIA

TOOLKIT IV

TURTLEGRAPHICS

Requires
Kyan Pascal (Version 2.0)

TABLE OF CONTENTS

Notice

Use of Routines in this Toolkit

Kyan Software hereby grants you a non-exclusive license to merge or use the routines in this Toolkit in conjunction with your own programs for either private or commercial purposes.

Copyright

This users manual and the computer software (programs) described in it are copyrighted by Kyan Software Inc. with all rights reserved. Under the copyright laws, neither this manual nor the programs may be copied, in whole or part, without the written consent of Kyan Software Inc. The only legal copies are those required in the normal use of the software or as backup copies. This exception does not allow copies to be made for others, whether or not sold. Under the law, copying includes translations into another language or format.

This restriction on copies does not apply to copies of individual routines copied and distributed as an integral part of programs developed by the purchaser of this Toolkit.

Backup Copies

We strongly recommend that you make and use backup copies of the Toolkit diskette. Keep your original Kyan diskettes in a safe location in case something happens to your copies. (Remember ..... Murphy is alive and well, and he loves to mess with computers!)

Copy Protection

Kyan Software products are not copy-protected. As a result, you are able to make backup copies and load your software onto a hard disk or into a RAM expansion card. We trust you. Please do not violate our trust by making or distributing illegal copies.

Limited Warranty

Kyan Software warrants the diskette(s) on which the Kyan software is furnished to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of delivery to you as evidenced by your proof of purchase.

Disclaimer of Warranty -- Kyan Software Inc.

Except for the limited warranty described in the preceding paragraph, Kyan Software makes no warranties, either express or implied, with respect to the software, its quality, performance, merchantability or fitness for any particular purpose. This software is licensed "as is". The entire risk as to its quality and performance is with the Buyer. Should the software prove defective following its purchase, the buyer (and not Kyan Software, its distributors, or its retailers) assumes the entire cost of all necessary servicing, repair, or correction and any incidental, or consequential damages.

In no event, will Kyan Software be liable for direct, or indirect, incidental, or consequential damages resulting from any defect in the software even if it has been advised of the possibility of such damages. The sole obligation of Kyan Software Inc. shall be to make available for purchase, modifications or updates made by Kyan Software to the software which are published within one year from date of purchase, provided the customer has returned the registration card delivered with the software.

Some states do not allow the exclusion or limitation of implied warranties or liabilities for incidental or consequential damages, so the above limitations or exclusions may not apply to you.

If any provisions or portions of this Agreement shall be held by a court of competent jurisdiction to be contrary to law, the remaining provisions of this Agreement shall remain in full force and effect. The validity, construction and performance of this Agreement shall be governed by the substantive law of the State of California.

This Agreement constitutes the entire agreement between the parties concerning the subject matter hereof.

Technical Support

Kyan Software has a technical support staff ready to assist you with any problems you might encounter. If you have a problem, we request that you first consult this users manual.

If you have a problem which is not covered in the manual, our support staff is ready to help. If the problem is a program which won't compile or run, we can best help if you send us a description of the problem and a listing of your program (better yet, send us a disk with the listing on it). We will do our best to get back to you with an answer as quickly as possible.

If you question can be answered on the phone, then give us a call. Our technical staff is available to assist on Monday through Friday between the hours of 9 AM and 5 PM, West Coast Time. You may reach them by calling:

Technical Support: (415) 626-2080

Suggestion Box

Kyan Software likes to hear from you. Please write if you have sugges-tions, comments and, yes, even criticisms of our products. We do listen. It is your suggestions and comments that frequently lead to new products and/or product modifications.

We encourage you to write. To make it easier, we have included a form in the back of this manual. This form makes it easier for you to write and easier for us to understand and respond to your comments. Please let us hear from you.

Mailing Address: **Kyan Software Inc.**
1850 Union Street #183
San Francisco, CA 94123

A. Introduction

Thank you for purchasing this TurtleGraphics Toolkit. It is designed for use with Kyan Pascal (Version 2.0 or later) and an Apple // with at least 64K of memory (RAM).

Overview

The Toolkit contains many useful and powerful routines which can be merged directly into your Kyan Pascal programs. These routines are grouped into three libraries or directories.

I. TurtleGraphics Library

This library contains routines which allow you to use TurtleGraphics in your programs. The library routines include:

o InitTurtle	o PenColor	o GrafMode
o TextMode	o Turn	o TurnTo
o Move	o MoveTo	o TurtleX
o TurtleY	o TurtleAng	o ViewPort
o FullPort	o FillPort	o SaveHires
o LoadHires		

II. Sound Effects Library

This library contains four procedure used to generate sound effects in an application program. The routines include:

o Beep	Rings the Apple bell
o Note	Sounds a tone with a specified pitch and duration
o Click	Generates a click from the speaker
o Phaser	Creates a phaser sound effect, a specified number of times

III. Chart Routines

This library contains 3 procedures which allow you to graphically display data. The routines include:

o BarChart	Draws proportioned bar graph of data
o PieChart	Generates a pie chart
o PlotXY	Plots an X vs. Y graph, point by point

How to Use the System Utilities

The routines in each Library are text files and are structured to be used as "include" files in your Pascal programs. To use them:

1. Copy the desired Toolkit routine(s) into your current working directory.
2. Declare the "included" file(s) in the declarations portion of your program.
3. Call the routine(s) as required in the body of your program.

Some libraries require global types to be separately declared. The steps for declaring these global types are described later in this Manual.

While most of the Toolkit routines are independent of all others, some routines incorporate others in the body of their programs. In these circumstances, it is necessary to include both Toolkit routines in your Pascal program. If a routine is dependent on some other routine, the dependency is noted in the application notes for the routine.

It is a good idea to review the section in Chapter III of your Kyan Pascal manual which describes the use of "include" files in your Pascal programs. You should also look at Chapter V which describes assembly language programming and Appendices C-F which list the meaning of MLI and other error messages.

You are encouraged to examine the source code of the Toolkit routines. To do so, simply load the routine's include file using the Kyan Text Editor. The source files are fully commented, and so you should be able to easily follow the logic and flow of the program. You can also modify any of the routines, if desired, and customize them for your particular application.

The Appendix illustrates the directory and file organization of the TurtleGraphics Toolkit disk. Always be sure to specify the complete pathname of the include file when you are copying routines into your working directory or running the demonstration programs. Also, when running the demo programs, be sure there is a copy of the Kyan Pascal Runtime Library (LIB) in the working directory.

Demonstration Programs

The TurtleGraphics Toolkit contains a number of demonstration programs which illustrate the use of Toolkit routines. Most of these programs are included in both source and object code formats.

TITLE	DESCRIPTION
TURTLEDEMO.P	This program illustrates the concepts of turtle angles, drawing polygons, drawing inside andoutside the viewport, filling irregular shapes, and provides samples of all 21 different hi-res fill patterns. Use this program as a demonstration for the TurtleGraphics utilities.
SOUND.DEMO.P	Shows how sound effects can dress up a program.
CHART.DEMO.P	Demonstrates the chart routines in conjunction with TurtleGraphics routines. Use this program as a guide when constructing the linked lists used as the source of display data. used with TurtleGraphics.

B. TurtleGraphics Library

Overview

The TurtleGraphics Library contains 17 different routines. They include a mix of functions and procedures which can be incorporated into your Pascal programs. Each TurtleGraphics routine is described on the following pages.

The TurtleGraphics routines in this Library include:

FILLAREA
FILLPORT
FULLPORT
GRAFMODE
INITTURTLE
LOADHIRES
MOVE
MOVETO
PENCOLOR
SAVEHIRES
TEXTMODE
TURN
TURNTO
TURTLEANG
TURTLEX
TURTLEY
VIEWPORT

TurtleGraphics

The basic concept behind "TurtleGraphics" is remarkably simple. Imagine a "turtle" facing some angle, dragging behind it a paint brush. As the turtle moves, it leaves behind a line in the same color as its brush. Apply this turtle to Pascal, interface the Apple's hi-resolution graphics capabilities, and you get the TurtleGraphics Library for Kyan Pascal.

The hi-res screen is made up of pixels. Each pixel can be thought of as a different place upon which the turtle may come to rest. Every pixel has an X and a Y co-ordinate. The X co-ordinates increase from 0 at the leftmost part of the screen to 279 at the rightmost part of the screen. Similarly, Y increases from 0 at the top of the screen to 191 at the bottom of the screen:

```
          0     <--------------X------------------>      279

              ___________________________________________
  0         |                                           |
            |                                           |
  ^         |                                           |
  |         |                                           |
  |         |                                           |
  Y         |                                           |
  |         |                                           |
  |         |                                           |
  V         |                                           |
 191        |                                           |
            |___________________________________________|
```

So if the turtle is resting at position (100,31), it is 100 pixels from the left of the screen and 31 pixels down from the top of the screen. If the turtle ever steps off the screen, you won't be able to see where it is (but it's there nonetheless). Its X and Y position can be negative or positive. The turtle trail only appears in the current "viewport". A viewport is the part of the hi-res screen which you specify the turtle's trail can be seen in. The view port doesn't have to be the entire hi-res screen; you can specify the use of only the left half, right half, or any other visible rectangle.

How does this relate to Pascal? The routines in the TurtleGraphics Library manipulate the turtle from within a Pascal program. In order to use these routines you must use the following lines of code at the beginning of your Pascal program:

```
#A
_UsesHires
#
```

Doing so tells the Pascal compiler that your program must reserve memory locations $2000 thru $3FFF for use by the hi-resolution graphics screen.

For a good example of the use of TurtleGraphics routines, be sure to read over the TURTLE.DEMO.P source file on your Utilities Diskette. In it you'll find routines for drawing polygons and circles; an example of how to use the viewport effectively; and,other useful procedures.

Using the TurtleGraphics Library

Two global types must be declared with the TurtleGraphics Library. To do this you must "include" the following files in your Pascal program using the following format:

```
Type
#i  TURTLE.TYPES.I
```

Then you must include the TurtleGraphics Library files:

```
Var
......(Declared variables)
#i  TURTLE.LIB.I
```

After you have included these files you can call the procedures you want to use from your program.

Command Name: ***Fill Area***

Syntax: PROCEDURE FILLAREA(x,y,PatternNumber: INTEGER);

Description: Fill area starting at x,y with corresponding pattern. (x,y) must be valid screen co-ordinates; the black background starting at (x,y) is filled until a non-black pixel is encountered in each of the four directions from the starting point. Note that more than one call to FILLAREA may be required to fill an irregular shape.

****

Command Name: ***Fill Port***

Syntax: PROCEDURE FILLPORT(fillcolor: ColorType);

Description: This procedure fills the current viewport with the color specified. The current pencolor is not changed, nor is the turtle's position.

****

Command Name: ***Full Port***

Syntax: PROCEDURE FULLPORT;

Description: This procedure sets the viewport to full screen: (0,0) through (279,191).

Command Name: ***Graph Mode***

Syntax: PROCEDURE GRAFMODE;

Description: This procedure displays the graphics page without changing it.

****

Command Name: ***Initialize Turtle***

Syntax: PROCEDURE INITTURTLE;

Description: This routine prepares the hi-res screen for use with the turtle by: clearing it to black; centering the turtle at co-ordinates (139,95); setting the turtle's angle to zero (facing right); setting PENCOLOR to "none"; making the viewport full screen (279 by 191); and, displaying the screen.

****

Command Name: ***Load Hires***

Syntax: FUNCTION LOADHIRES(VAR pathname: PathString):
INTEGER;

Description: This function loads a binary image starting at $2000 using the pathname passed. Any BIN file can be used, and only the first $2000 bytes are actually moved into memory. The function value returned is the resulting MLI error code. **NOTE:** This function requires the following global type declaration:

```
PathString = ARRAY[1..65] of CHAR;
```

Command Name: ***Move***

Syntax: PROCEDURE MOVE(distance: INTEGER);

Description: This procedure moves the turtle 'distance' pixels in the current TurtleAngle direction. Only the parts of the resulting line which appear in the viewport will be displayed. If PENCOLOR is "none", no actual drawing will take place, but the turtle will have moved anyway.

****

Command Name: ***Move To***

Syntax: PROCEDURE MOVETO(x,y: INTEGER);

Description: This routine tells the turtle to move to screen co-ordinates (x,y). Only the parts of the resulting line which appear in the viewport will be displayed. If PENCOLOR is "none", no actual drawing will take place, but the turtle will have moved anyway.

****

Command Name: ***Pen Color***

Syntax: PROCEDURE PENCOLOR(color: ColorType);

Description: This procedure tells the turtle what color to "drag" behind him as he moves. If you specify NONE, the turtle leaves no trail when he moves. You must use one of the names declared in the global types declarations in the TURTLE.TYPES.I file (i.e., white, green, violet, red, blue, black, white1, black1, none).

Command Name: ***Save Hires***

Syntax: FUNCTION SAVEHIRES(VAR pathname: PathString):
INTEGER;

Description: This function writes the current hi-res image starting at $2000 using the pathname passed. **NOTE:** This function requires the following global type declaration.

```
PathString = ARRAY[1..65] of CHAR;
```

****

Command Name: ***Text Mode***

Syntax: PROCEDURE TEXTMODE;

Description: This procedure displays text page 1 without changing it.

****

Command Name: ***Turn***

Syntax: PROCEDURE TURN(angle: INTEGER);

Description: This procedure turns the turtle "angle" degrees. The resulting angle is always between 0 and 359 degrees (0 faces right). You may TURN the turtle a negative number of degrees; doing so turns the turtle in a clockwise direction.

Command Name: ***Turn To***

Syntax: PROCEDURE TURNTO(angle: INTEGER);

Description: This procedure points the turtle at "angle" degrees, without regard for it's previous angle. Since 'angle' is always between 0 and 359, the number you use may be negative and is treated as if the turtle moved that number of degrees clockwise (down and to the right).

****

Command Name: ***Turtle Angle***

Syntax: FUNCTION TURTLEANG:INTEGER;

Description: This function returns the angle the turtle is currently facing (0-359) in degrees.

****

Command Name: ***Turtle X***

Syntax: FUNCTION TURTLEX: INTEGER

Description: This function returns the current position of turtle's X co-ordinates.

Command Name: ***Turtle Y***

Syntax: FUNCTION TURTLEY: INTEGER;

Description: This function returns the current position of turtle's Y co-ordinates.

****

Command Name: ***View Port***

Syntax: PROCEDURE VIEWPORT(xmin, xmax, ymin, ymax:
INTEGER);

Description: This procedure limits the turtle's drawing screen to the co-ordinates passed. If the screen values specified are not valid screen positions, the command is ignored.

C. Chart Library

Overview

The Chart Library consists of three different procedures. They are used in conjunction with the TurtleGraphic routines described in section B.

The Chart routines are:

BARCHART	: Draws a bar chart
PIECHART	: Draws a pie chart
X-YPLOT	: Plots a graph

Using the Chart Library

To use the routines in the Chart Library, you must first declare a set of global types and then "include" the desired Chart Library file after the variable and type declarations in your Pascal program. Once the libraries are included, the routines can be called as often as needed in your program, just as you would call a procedure. In order to use any of the Chart routines, the TurtleGraphics library and Turtle Graphic global types must also be included. The declarations should look as follows:

```
Type
#i  CHART.TYPES.I
#i  TURTLE.TYPES.I

Var
...........( Declared variables )
#i  CHART.LIB.I
#i  TURTLE.LIB.I
```

NOTE: Please refer to Chapter III of the Kyan Pascal Manual for more information about the use of include files in Pascal programs.

Command Name: ***Bar Chart***

Syntax: PROCEDURE BARCHART(BasePtr: GrRecPtr);

Description: This procedure draws a simple bar chart. The bar size is based on the number of nodes on the data list and the range of their values. The bars are drawn in alternating hi-res colors. More than 5 bars will repeat the colors. An empty list will result in a blank screen. The routine ends with the viewport full and hi-res screen displayed.

****

Command Name: ***Pie Chart***

Syntax: PROCEDURE PIECHART(BasePtr: GrRecPtr);

Description: This procedure draws a pie chart. The data in each node is compared to the total of the data fields on the data list. Then each 'slice' of the pie is a percentage of the total of the data fields. This routine ends with the viewport full and the hi-res screen displayed.

****

Command Name: ***XY Plot***

Syntax: PROCEDURE XYPLOT(BasePtr: GrRecPtr;
DrawAxes,ConnectPoints: BOOLEAN);

Description: This procedure draws an "X versus Y" graph. The scaling for each axis is determined by examining the data on the data list. If you want to display the X and Y axes, then set the DrawAxes variable to TRUE; if not,, set DrawAxes to FALSE. If you want to connect each point in the plot with a white line, then set ConnectPoints to TRUE; if not, set ConnectPoints to FALSE. This routine exits with the viewport full and the hi-res screen displayed.

D. Sound Effects Library

Overview

The Sound Effects Library contains four different routines. These routines are:

BEEP	: Sound the Apple 'beep'
CLICK	: Generate a click from the speaker
NOTE	: Generate a note of specific pitch and duration
PHASER	: Generate a pulsing tone

Using the Sound Effects Library

To use the routines in the Sound Effects Library, you must first "include" SOUND.LIB.I after the variable declarations in your Pascal program. Then call the routines you want to use in your program as you would a procedure. Declaring global types associated with the Sound Effects Library is not necessary. (NOTE: Please refer to Kyan Pascal Manual Chapter III for more information about the use of "include" files in Pascal programs)

****

Command Name: ***Beep***

Syntax: PROCEDURE BEEP;

Description: This procedure has a ringing bell sound.

****

Command Name: ***Click***

Syntax: PROCEDURE CLICK;

Description: This procedure generates a very brief "clicking" sound from the speaker.

Command Name: ***Note***

Syntax: PROCEDURE NOTE(pitch, duration: INTEGER);

Description: This procedure generates a note according to "pitch" for "duration" time. The approximate "pitch" values for the standard music scale are:

"High"	C	48
	B	51
	Bb	54
	A	57
	Ab/G#	60
	G	64
	F#	68
	F	72
	E	76
	Eb/D#	80
	D	85
	Db/C#	90
"Middle"	C	96
	B	102
	Bb	108
	A	114
	Ab/G#	121
	G	128
	F#	136
	F	144
	E	153
	Eb/D#	161
	D	171
	Db/C#	181
"Low"	C	192
	B	204
	Bb	216
	A	228
	Ab/G#	242
	G	255

Command Name: ***Phaser***

Syntax: PROCEDURE PHASER(pulse: INTEGER);

Description: This procedure generates a "phaser" effect "pulse" number of times. This routine requires the NOTE procedure and so it must be included in the Pascal program.

E. APPENDIX DISK DIRECTORY

Volume Name:	TURTLEGRAPHICS.TOOLKIT	
Include Files:	TURTLE.TYPES.I (Global Types)	
	TURTLE.LIB.I	
Procedures:		FILLAREA
		FILLPORT
		FULLPORT
		GRAFMODE
		INITTURTLE
		LOADHIRES
		MOVE
		MOVETO
		PENCOLOR
		SAVEHIRES
		TEXTMODE
		TURN
		TURNTO
		TURTLEANG
		TURTLEX
		TURTLEY
		VIEWPORT
Include Files:	CHARTS.TYPES.I (Global Types)	
	CHART.LIB.I	
Procedures:		BARCHART
		PIECHART
		XYPLOT
Include Files:	SOUND.LIB.I	
Procedures:		BELL
		NOTE
		CLICK
		PHASER

Directory: UTL.DEMOS

CHART.DEMO.P (Source)
CHART.DEMO (Object)

SOUND.DEMO.P (Source)
SOUND.DEMO (Object)

TURTLE.DEMO.P (Source Code)
TURTLE.DEMO (Object Code)

Suggestion Box

We do our best to provide you with complete, bug-free software and documentation. With products as complex as compilers and programming utilities, this is difficult to do. If you find any bugs or areas where the documentation is unclear, please let us know. We will do our best to correct the problem in the next revision. We would also like to hear from you if have any comments or suggestions regarding our product.

To help us better understand your comments please use the following form in your correspondence and mail it to: Kyan Software Inc., 1850 Union Street #183, San Francisco, CA 94123.

Name____
Address____
City__________________________State______ZIP__________
Telephone:
(day)______________________ (evening)__________________

Kind of Problem	**Software Description**
__ Software Bug	Product Name ______________
__ Documentation Error	Version No. ________________
__ Suggestions	Date Purchased ______________
__ Other ______________________	

Kyan Software Products You Use

__ Kyan Pascal	__ Kyan Macro Assembler/Linker
__ System. Utilities Toolkit	__ Advanced Graphics Toolkit
__ MouseText Toolkit	__ MouseGraphics Toolkit
__ TurtleGraphics Toolkit	__ Other ________________

Your Hardware Configuration

Type/Model of Computer ______________________________
How many and what kind of disk drives __________________
What is your screen capability: ___40 Column ___80 Column
How much RAM?____K (what kind of RAM Board?______________)
What kind of printer and interface card do you use?______________
____
What kind of modem?__________________________________
Other information about your computer system: _______________
____

What do you use this software for?

___ Education (I am a __ teacher __ student)
___ Hobby
___ Professional Software Development
___ Other ____

Problem Description (if appropriate, please include a disk or program listing).

Suggestions

TI 8605A